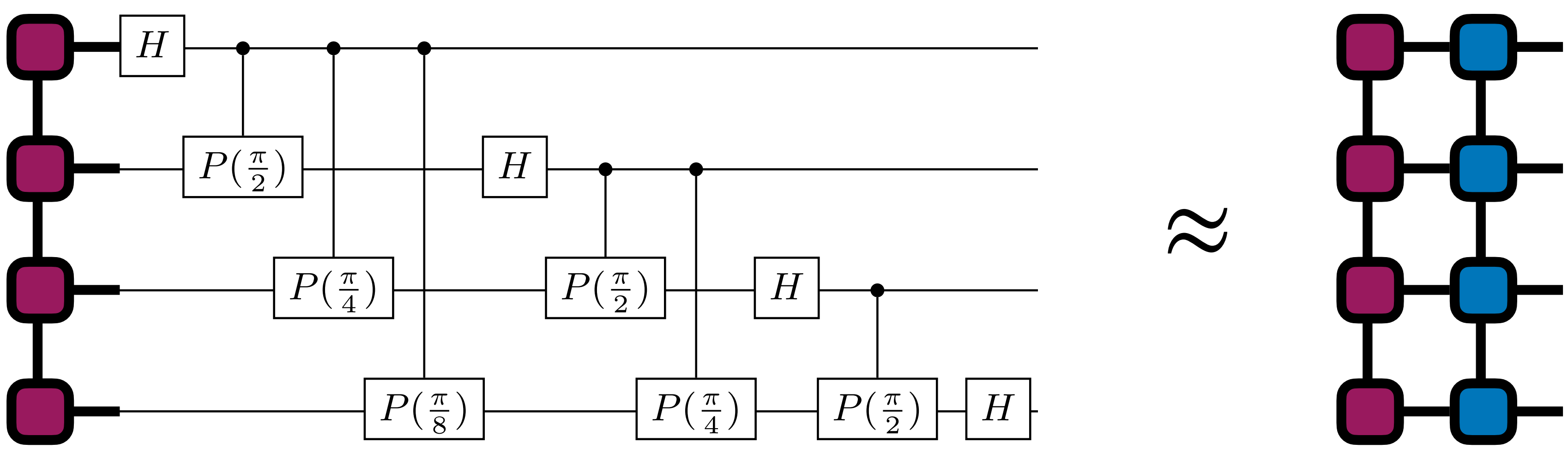
ITensor MPS and MPO Algorithms



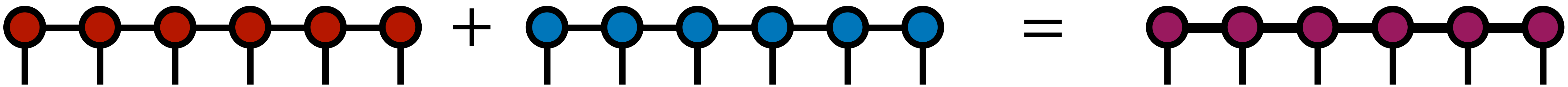
E.M. Stoudenmire

August 2024 - U. Mass. Amherst

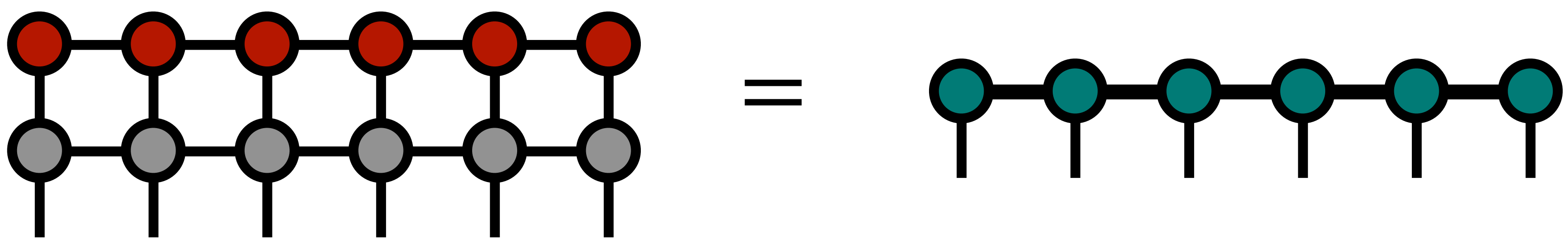
FLATIRON INSTITUTE
SIMONS FOUNDATION

Matrix Product State (MPS) Algorithms

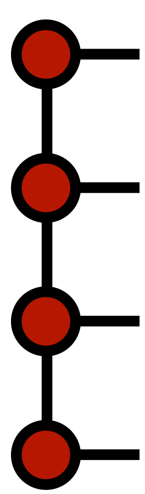
Can efficiently sum MPS in compressed form:



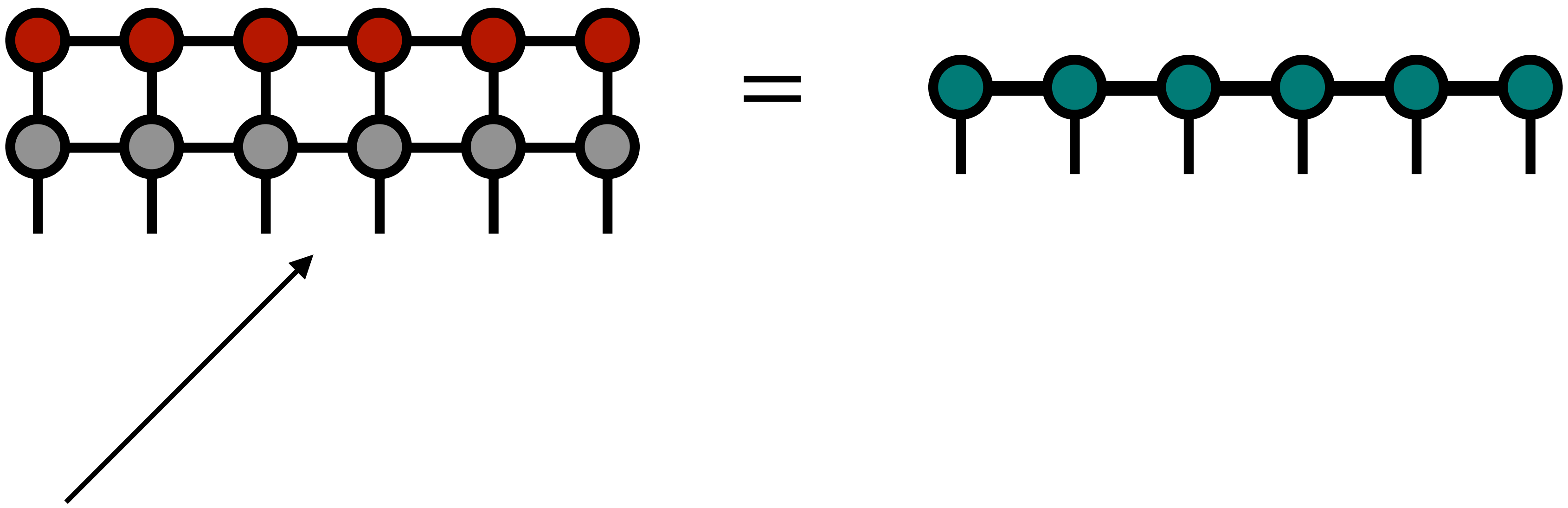
multiply by other networks:



and perfectly sample:



0	0	0	0	1	0
0	1	1	0	1	0
0	1	0	1	0	0
0	0	0	0	1	0



matrix product operator (MPO) network

MPS Algorithms

We offer MPS and "MPO" algorithms through the
ITensorMPS
package

```
using ITensors
using ITensorMPS

sites = siteinds("S=1/2",N)

terms = OpSum()
for j=1:N-1
  terms += "Sz",j, "Sz",j+1
end

H = MPO(terms,sites)
```

MPS Algorithms

The ITensorMPS package offers many helpful algorithms for working with MPS and MPO's

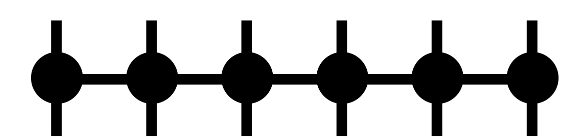
- OpSum system – making MPOs from operators
- dmrg & tdvp – computing ground states and dynamics
- expect – computing expected values of operators
- correlation_matrix – compute correlation functions
- inner – overlap MPS and MPOs
- contract, sum – algebra of MPS and MPO

MPS Algorithms

OpSum – powerful "domain-specific language" (DSL)
for making MPOs from math expressions

$$\sum_j S^z_j S^z_{j+1} \longrightarrow$$

```
sites = siteinds("S=1/2",N)

terms = OpSum()
for j=1:N-1
  terms += "Sz",j, "Sz",j+1
end

H = MPO(terms,sites)
```

$\longrightarrow$  H

MPS Algorithms

Starting from beginning: first make an array of "sites"
Just a Julia array of Index objects

```
sites = siteinds("S=1/2",N)
```

MPS Algorithms

Next fill up OpSum with "terms" of the operator

```
sites = siteinds("S=1/2",N)

terms = OpSum()
for j=1:N-1
  terms += "Sz",j, "Sz",j+1
end
```

Internal data of "terms" similar to:

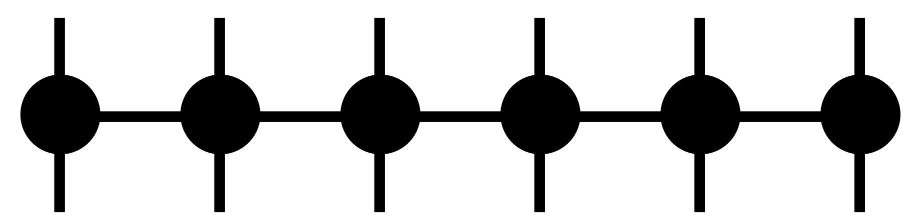
(1.0,"Sz",1,"Sz",2),   (1.0,"Sz",2,"Sz",3), ...

MPS Algorithms

Finally, construct MPO – compresses terms together [1,2]

```
sites = siteinds("S=1/2",N)

terms = OpSum()
for j=1:N-1
  terms += "Sz",j, "Sz",j+1
end

H = MPO(terms,sites)
```

(1.0,"Sz",1,"Sz",2), (1.0,"Sz",2,"Sz",3), ...

$H =$ 

Often optimal "bond dimension" reached

[1] Chan, Keselman, Nakatani, Li, White, J. Chem. Phys 145 (2016)
[2] Parker, Zaletel, Phys. Rev. B, 102, 035147 (2020)

MPS Algorithms

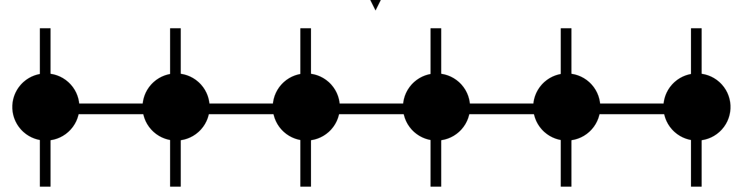
Very wide range of operators can be made

$$H = \sum_j S^z_j S^z_{j+1}$$

$\downarrow$

```julia
sites = siteinds("S=1/2",N)

terms = OpSum()
for j=1:N-1
  terms += "Sz",j, "Sz",j+1
end

H = MPO(terms,sites)
```

$\downarrow$

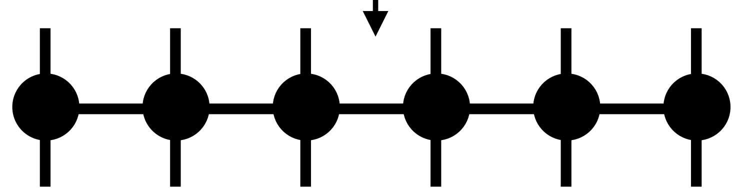
$H =$ 

MPS Algorithms

Very wide range of operators can be made

$$H = \sum_j S^z_j S^z_{j+1} + \frac{1}{2} S^+_j S^-_{j+1} + \frac{1}{2} S^-_j S^+_{j+1}$$

```
sites = siteinds("S=1/2",N)

terms = OpSum()
for j=1:N-1
  terms +=       "Sz",j, "Sz",j+1
  terms += 1/2,"S+",j, "S-",j+1
  terms += 1/2,"S-",j, "S+",j+1
end

H = MPO(terms,sites)
```

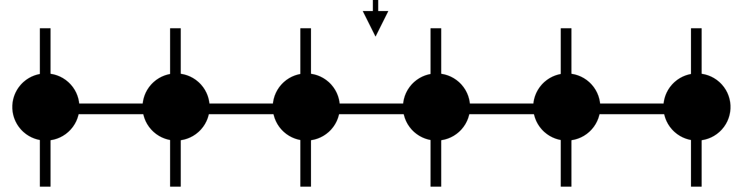
$H =$ 

MPS Algorithms

Changing site type automatically gives correct operators

$$H = \sum_j S^z_j S^z_{j+1} + \frac{1}{2} S^+_j S^-_{j+1} + \frac{1}{2} S^-_j S^+_{j+1}$$

```
sites = siteinds("S=1/2",N)

terms = OpSum()
for j=1:N-1
  terms +=       "Sz",j, "Sz",j+1
  terms += 1/2,"S+",j, "S-",j+1
  terms += 1/2,"S-",j, "S+",j+1
end

H = MPO(terms,sites)
```

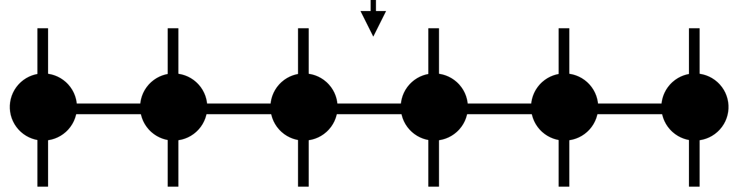
$H =$ 

MPS Algorithms

Changing site type automatically gives correct operators

$$H = \sum_j S^z_j S^z_{j+1} + \frac{1}{2} S^+_j S^-_{j+1} + \frac{1}{2} S^-_j S^+_{j+1}$$

```
sites = siteinds("S=1",N)

terms = OpSum()
for j=1:N-1
  terms +=       "Sz",j, "Sz",j+1
  terms += 1/2,"S+",j, "S-",j+1
  terms += 1/2,"S-",j, "S+",j+1
end

H = MPO(terms,sites)
```

$H =$ 

MPS Algorithms

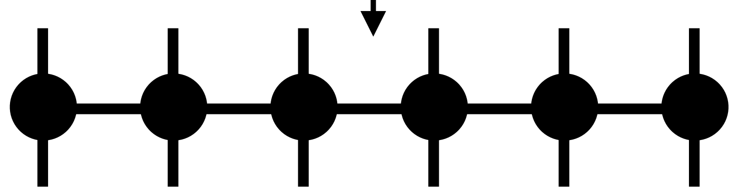
Particles (bosons, fermions) possible too

$$H = \sum_j c_j^\dagger c_{j+1} + c_{j+1}^\dagger c_j$$

$$\downarrow$$

```
sites = siteinds("Fermion",N)

terms = OpSum()
for j=1:N-1
  terms += "Cdag",j, "C",j+1
  terms += "C",j+1, "Cdag",j+1
end

H = MPO(terms,sites)
```

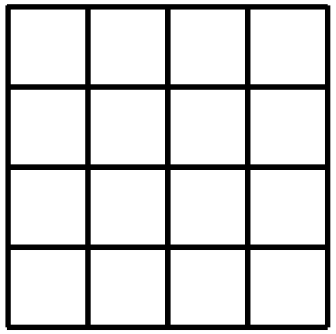
$$\downarrow$$

$H =$ 

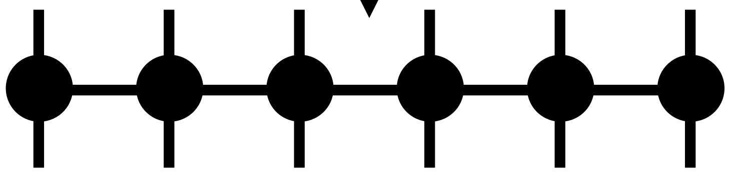
MPS Algorithms

And quasi-two-dimensional systems

$$H = \sum_{\langle ij \rangle} S^z_i S^z_j + \frac{1}{2} S^+_i S^-_j + \frac{1}{2} S^-_i S^+_j$$



```
lattice = square_lattice(Nx, Ny; yperiodic=false)

terms = OpSum()
for b in lattice
  terms +=      "Sz", b.s1, "Sz", b.s2
  terms += 1/2, "S+", b.s1, "S-", b.s2
  terms += 1/2, "S-", b.s1, "S+", b.s2
end

H = MPO(terms, sites)
```

$H =$ 

MPS Algorithms

Next let's look at dmrg and tdvp

- [x] OpSum system – making MPOs from operators
- [ ] dmrg & tdvp – computing ground states and dynamics
- [ ] expect – computing expected values of operators
- [ ] correlation_matrix – compute correlation functions
- [ ] inner – overlap MPS and MPOs
- [ ] contract, sum – algebra of MPS and MPO

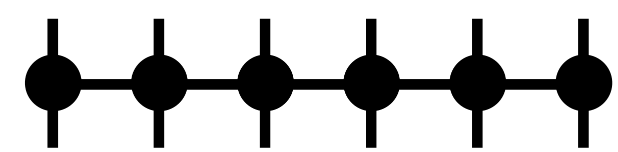
MPS Algorithms

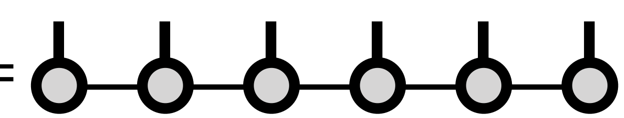
ITensorMPS offers "black box" DMRG algorithm

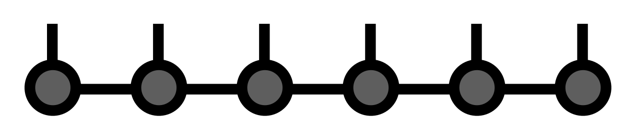
```julia
using ITensors, ITensorMPS

N = 100
sites = siteinds("S=1", N)

terms = OpSum()
for j in 1:(N - 1)
  terms += "Sz", j, "Sz", j + 1
  terms += 0.5, "S+", j, "S-", j + 1
  terms += 0.5, "S-", j, "S+", j + 1
end
H = MPO(terms, sites)

psi0 = random_mps(sites; linkdims=10)

nsweeps = 5
maxdim = [10, 20, 100, 100, 200]
cutoff = [1E-11]
energy, psi = dmrg(H, psi0; nsweeps, maxdim, cutoff)
```

$$H = \sum_j S^z_j S^z_{j+1} + \frac{1}{2} S^+_j S^-_{j+1} + \frac{1}{2} S^-_j S^+_{j+1}$$

(1.0,"Sz",1,"Sz",2), (1.0,"Sz",2,"Sz",3), ...

$H =$ 

$\psi_0 =$ 

$\psi =$ 

MPS Algorithms

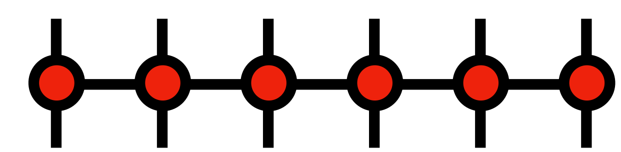
TDVP is similar, for time evolution by some H

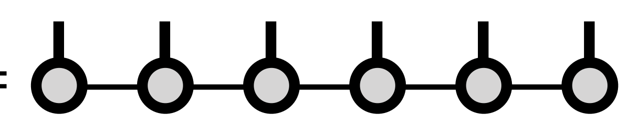
```
using ITensors, ITensorMPS

N = 100
sites = siteinds("S=1/2", N)

terms = OpSum()
for j in 1:(N - 1)
  terms += "Sz", j, "Sz", j + 1
  terms += 0.5, "S+", j, "S-", j + 1
  terms += 0.5, "S-", j, "S+", j + 1
end
H = MPO(terms, sites)

psi0 = random_mps(sites; linkdims=10)

t = 10
time_step = 0.1
maxdim = 200
cutoff = 1E-10
psi_t = tdvp(H, -im*t, psi0; maxdim, cutoff, time_step)
```

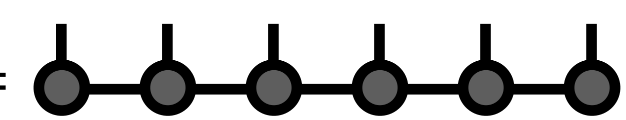
👀

$$H = \sum_j S^z_j S^z_{j+1} + \frac{1}{2} S^+_j S^-_{j+1} + \frac{1}{2} S^-_j S^+_{j+1}$$

(1.0,"Sz",1,"Sz",2), (1.0,"Sz",2,"Sz",3), ...

$e^{-iHt} =$ 

$\psi(0) =$ 

$\psi(t) =$ 

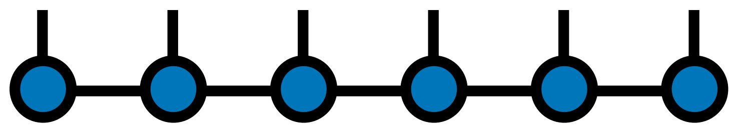
MPS Algorithms

Can use expect, correlation_matrix, and inner to analyze MPS

- [x] OpSum system – making MPOs from operators
- [x] dmrg & tdvp – computing ground states and dynamics
- [ ] expect – computing expected values of operators
- [ ] correlation_matrix – compute correlation functions
- [ ] inner – overlap MPS and MPOs
- [ ] contract, sum – algebra of MPS and MPO

MPS Algorithms

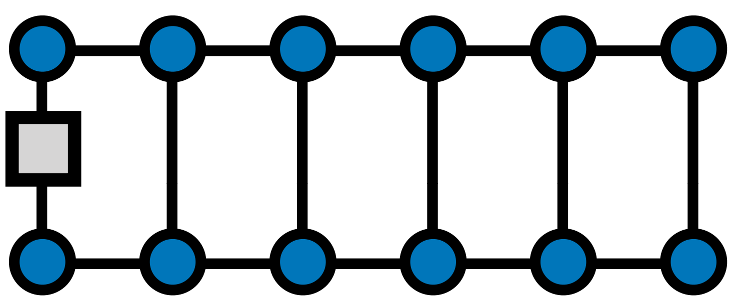
If we have an MPS and want expected values of local operators, can use expect function

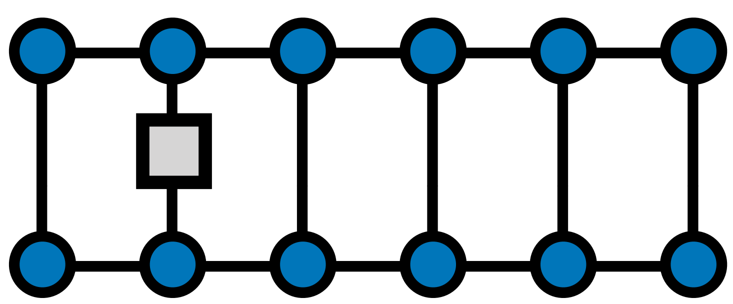
$\Psi =$  could be output of DMRG, TDVP

Then, for example, calling expect gives

```
sz = expect(psi,"Sz")
```

where sz is an array such that

sz[1] = 

sz[2] = 

etc.

MPS Algorithms

Of course, can use various operators for spins, particles, or qubit sites

Some examples:

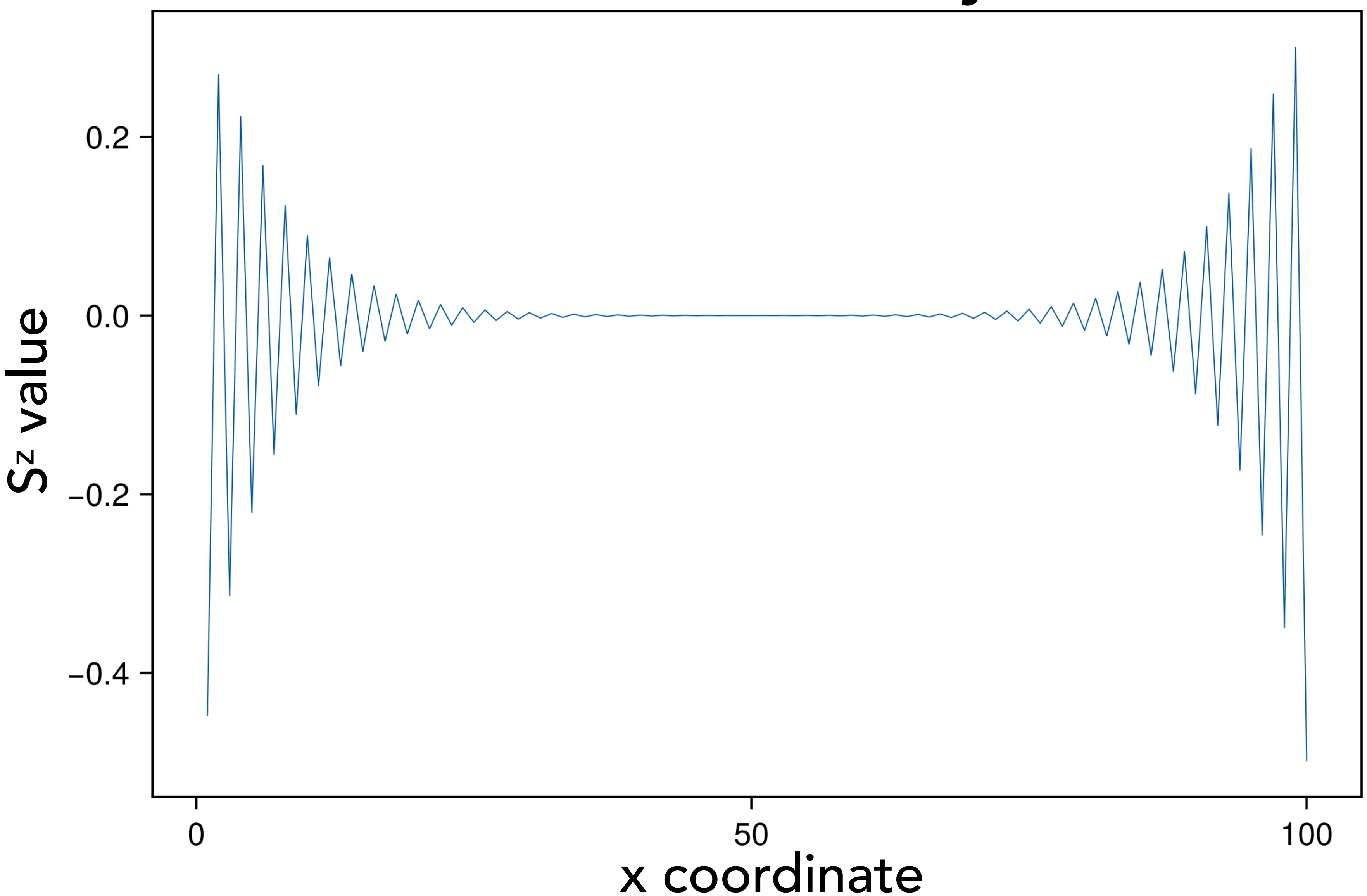
`sz = expect(psi,"Sz")`	magnetization of spins
`density = expect(psi,"N")`	density of particles
`Xvals = expect(psi,"X")`	<X> over a set of qubits

MPS Algorithms

Example output:

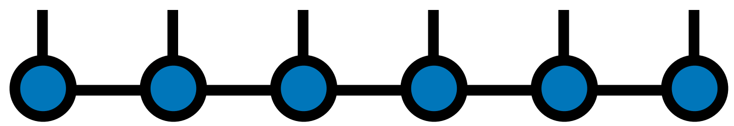
```
energy, psi = dmrg(H, psi0; nsweeps, maxdim, cutoff)
sz = expect(psi,"Sz")
```

Plot of sz array



MPS Algorithms

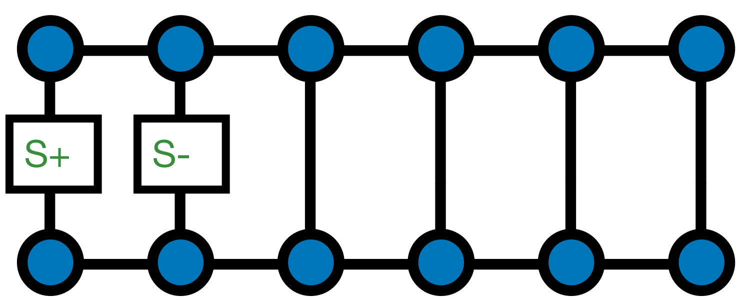
The correlation_matrix function also computes expected values but of "two point" functions i.e. correlators

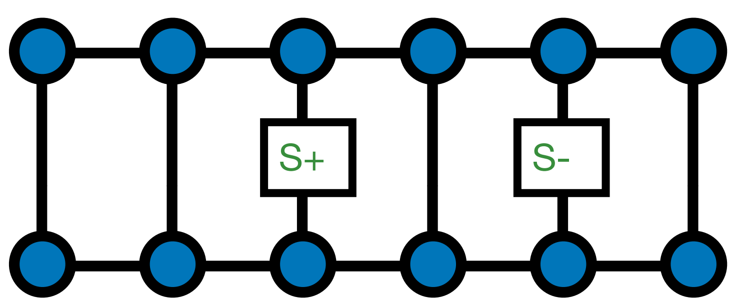
$\Psi =$  could be output of DMRG, TDVP

Then, calling correlation_matrix gives

```
spm = correlation_matrix(psi,"S+","S-")
```

where spm is an Matrix such that

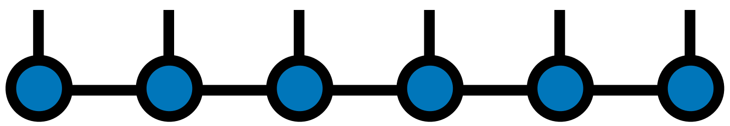
spm[1,2] = 

spm[3,5] = 

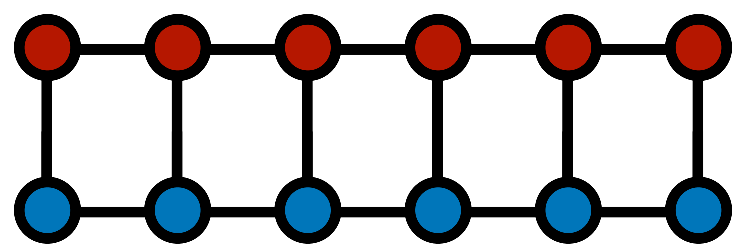
etc.

MPS Algorithms

The inner function lets us analyze MPS through overlaps with other MPS and MPOs

$\Psi =$  could be output of DMRG, TDVP

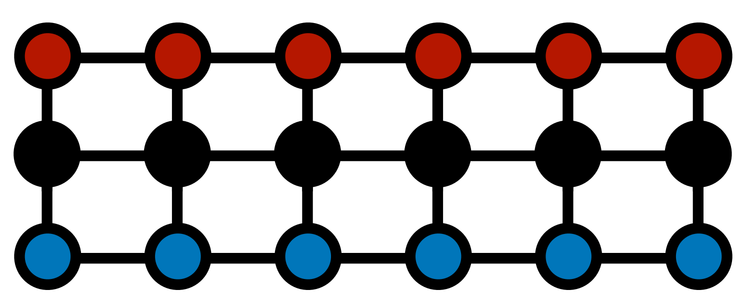
Then, calling inner with another MPS gives

inner(psi,phi) = 

Φ

Ψ

Or including an MPO like

inner(phi',H,psi) = 

Φ

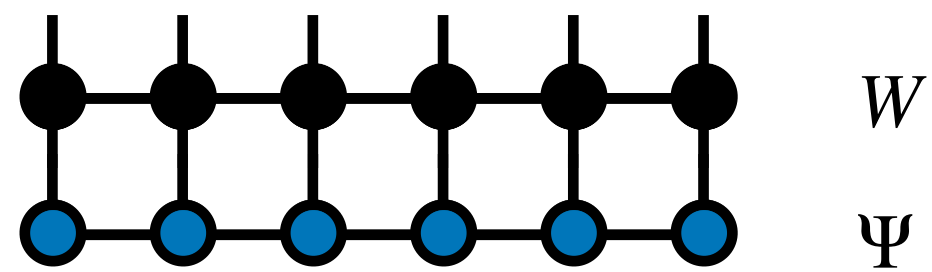
H

Ψ

MPS Algorithms

Finally MPS and MPO tensor networks can be contracted and added with contract and sum

contract(W,psi; cutoff) =



sum(psi,phi; cutoff) =